VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2014

Vojtěch Koval

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Vojtěch Koval**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Allegro Group CZ, s.r.o.
2. Struktura závěrečné zprávy:
a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c) Zvolený postup řešení zadaných úkolů
d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **RNDr. Eliška Ochodková, Ph.D.**

Konzultant bakalářské práce: Ing. Radka Radecká

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2.dubna 2014 .....

Rád bych zde poděkoval společnosti Allegro Group CZ, s.r.o. za možnost absolvování odborné praxe. Dále především Ing. Radce Radecké za příjemnou spolupráci a také RNDr. Elišce Ochodkové, Ph.D. za vedení mé práce.

Abstrakt

Cílem této bakalářské práce je popsat průběh mého působení ve společnosti NetDirect s.r.o., jež je členem společnosti Allegro Group CZ, s.r.o. Práce je rozdělena do čtyř logických oddílů. V první části je představena samotná společnost včetně mého pracovního zařazení v rámci odborné praxe. Dále jsou uvedeny a blíže specifikovány konkrétní úkoly, na nichž jsem se podílel. U jednotlivých úkolů jsou taktéž zmíněny i postupy a technologie, které jsem v průběhu jejich řešení využil. V předposlední části jsou shrnuty znalosti, které jsem během praxe uplatnil, včetně těch, které mi naopak chyběly. Závěr práce je vyčleněn pro zhodnocení dosažených výsledků a také pro celkové hodnocení odborné praxe.

Klíčová slova: .NET, C#, XSLT, MVC, Allegro Group, NetDirect, Odborná praxe, Scrum, Unit testování, Správa zdrojového kódu

Abstract

The aim of this bachelor thesis is to describe my work in NetDirect Ltd., a member of the company Allegro Group CZ, Ltd. The thesis is divided into four logical parts. The first part describes the company itself, including my job position in the context of professional practice. Next there are listed and more closely specified tasks in which I was participating. For every task are also mentioned the procedures and technologies that I was using during their solution. The penultimate section summarizes the knowledge that I was applying during the practice, including those, which I missed. The conclusion of the thesis is set apart for the evaluation of achieved results and also for the whole evaluation of professional practice.

Keywords: .NET, C#, XSLT, MVC, Allegro Group, NetDirect, Professional practice, Scrum, Unit testing, Source code management

Seznam použitých zkratek a symbolů

ASCII	–	American Standard Code for Information Interchange
ASP	–	Active Server Pages
BL	–	Business Logic
CDATA	–	Character Data
CMS	–	Content Management System
CSS	–	Cascading Style Sheets
CVS	–	Concurrent Version System
DLL	–	Dynamic Link Library
ERM	–	Entity Relationship Model
ERP	–	Enterprise Resource Planning
GUID	–	Globally Unique Identifier
HTML	–	HyperText Markup Language
HTTP	–	HyperText Transfer Protocol
ID	–	Identification
IIS	–	Internet Information Services
JSON	–	JavaScript Object Notation
LINQ	–	Language Integrated Query
MVC	–	Model View Controller
ORM	–	Object-Relational Mapping
PHP	–	Hypertext Preprocessor
SEO	–	Search Engine Optimization
SIM	–	Subscriber Identity Module
SQL	–	Structured Query Language
T-SQL	–	Transact-SQL
TFS	–	Team Foundation Server
URL	–	Uniform Resource Locator
USA	–	United States of America
WCF	–	Windows Communication Foundation
XML	–	Extensible Markup Language
XSL	–	Extensible Stylesheet Language
XSLT	–	XSL Transformations

Obsah

Seznam obrázků

1 Úvod

Rozhodl jsem se pro absolvování bakalářské práce ve formě odborné praxe, protože jsem chtěl uplatnit a využít teoretické znalosti a dovednosti nabyté během studia. Rovněž jsem se chtěl tímto způsobem blíže seznámit se skutečnými postupy používanými v praxi. Ačkoliv jsem již v minulosti měl možnost pracovat v malém vývojovém týmu, chtěl jsem si vyzkoušet práci i ve velké softwarové společnosti, kde bych spolupracoval společně s lidmi ze stejného odvětví. A právě díky odborné praxi se mi tato myšlenka splnila a měl jsem možnost získat spoustu cenných zkušeností z reálného prostředí. Taktéž jsem měl možnost vypozorovat i své nedostatky (např. při práci v kolektivu), které jsem se snažil v průběhu praxe eliminovat.

Na základě předchozích doporučení jsem si vybral společnost NetDirect s.r.o. (dále jen společnost), kde jsem byl po absolvování přijímacího pohovoru a vstupních testů přijat a zařazen do vývojového oddělení.

V následujících kapitolách budu popisovat náplň a průběh mé praxe ve společnosti. Nejprve představím samotnou společnost včetně jejích hlavních produktů. Dále popíšu své pracovní zařazení a architekturu projektu (jeho rozvrstvení a logickou strukturu). Poté se budu věnovat strategii použité při vývoji (Daily meeting[1], Sprint[2] atd.) a metodice pro řízení projektů nazývané Scrum. Následně blíže specifikuji veškeré úkoly, na nichž jsem se během praxe podílel, včetně postupů a technologií, které jsem při jejich řešení využil. Načež také shrnu uplatněné a chybějící znalosti. A v závěru práce poté uvedu subjektivní zhodnocení veškerých dosažených výsledků a zkušeností, které mi odborná praxe přinesla.

[1] Každodenní setkání týmu

[2] Období iterace trvající 2-4 týdny

2 O společnosti

Společnost NetDirect s.r.o. vznikla v roce 2002 v Ostravě a v současné době patří mezi nejvýznamnější dodavatele e-business aplikací nejen na českém, ale i na slovenském trhu. Zabývá se především tvorbou a vývojem aplikací pro podnikání na internetu (internetové obchody, redakční systémy a webové prezentace).

V srpnu roku 2011 se společnost stala součástí nizozemské společnosti MIH Allegro BV, která v rámci holdingu Allegro Group provozuje nejvýznamnější e-commerce[3] služby ve střední a východní Evropě. Mateřská MIH Allegro Group je mezinárodni e-commerce internetová společnost, která náleží do skupiny globální technologické korporace Naspers. Ta patří v současnosti k jedné z největších technologických firem na světě a její aktivity sahají až do sociální sítě Facebook nebo čínského gigantu Tencent[4].

Koncem března roku 2013 proběhla fúze společnosti NetDirect s.r.o. a skupiny Allegro Group CZ. Skupina Allegro Group CZ, s.r.o. vznikla v roce 2011 a soustředí se na internetový trh s transakcemi a webovými službami. Díky kterým mohou uživatelé na internetu současně prodávat i nakupovat na jednom místě – k čemuž mohou využít jimi nabízených platforem jako je Aukro či Heuréka a nebo si mohou zřídit e-shop právě pomocí služeb NetDirectu.

Obrázek 1: Logo společnosti NetDirect s.r.o.

2.1 Získaná ocenění

NetDirect s.r.o. staví svá softwarová řešení výhradně na technologiích společnosti Microsoft. S touto společností intenzivně spolupracuje již od svého vzniku. Od roku 2003 byl NetDirect držitelem titulu *Microsoft Certified Partner* a v letech 2007 až 2011 byl dokonce držitelem prestižního titulu *Microsoft Gold Certified Partner*. Společnost se také umístila na předních místech v řadě soutěží – získala první tři příčky na světové konferenci *WPC09* v USA a nejvyšší ocenění na partnerské konferenci *WPC10*, kde byla vyhlášena nejlepším partnerem roku a získala titul *2010 Microsoft Country Partner of the Year*.

[3] Elektronické obchodování
[4] Investiční holdingová společnost

V roce 2011 společnost vybojovala také první místo v kategorii *Vývoj software a webových aplikací – Webové řešení* a získala ocenění *Partner of the Year Software Development Finalist 2011*. V současnosti je NetDirect držitelem kompetencí *Gold Web Development*, *Silver Software Development* a *Silver Web Development*.

2.2 Hlavní produkty

Mezi nejvýznamnější produkty, které společnost NetDirect nabízí svým zákazníkům, patří internetové obchody FastCentrik a ShopCentrik a redakční systém MediaCentrik. FastCentrik je „krabicové" řešení e-shopu určené pro menší obchodníky, kteří žádají komplexní systém za rozumnou cenu. ShopCentrik je naopak řešení dodávané zákazníkům na klíč od analýzy, grafiky až po nasazení do ostrého provozu. Posledním významným produktem je CMS MediaCentrik, pomocí kterého lze provozovat webové stránky či tzv. „microsites" neboli speciální webové doplňky.

Kromě těchto produktů nabízí NetDirect svým zákazníkům ještě i specializované aplikace jako AppCentrik, která nabízí přehledný výběr modulů a doplňkových služeb. A ThemeCentrik, která naopak představuje databázi grafických šablon pro e-shopy.

2.3 Pracovní zařazení

Na základě absolvování přijímacího pohovoru a vstupních testů jsem byl přijat a následně zařazen do vývojového týmu, který se zabýval implementací nových funkcionalit pro produkt FastCentrik 2.0. Mou vedoucí se stala Ing. Radka Radecká, která mě postupně seznámila s náplní mé práce a s interními procesy.

V prvních dnech jsem byl zasvěcen do firemní struktury a poté zařazen do Maintenance[5] týmu, kde jsem pomáhal při opravách chyb a požadavků od zákazníků. Následně jsem dostal za úkol vyvinout tzv. Kuchařku neboli webovou aplikaci, která se skládala ze dvou částí – v první z nich byla přibližná struktura systému FastCentrik 2.0 včetně ukázek částí kódu, ve druhé byly naopak veškeré interní předpisy, které museli partneři a zejména designéři (či grafická studia) při tvorbě šablon pro tento systém striktně dodržovat.

[5] Servisní oddělení

3 Architektura projektu

Již zmíněný produkt FastCentrik 2.0 je založen na koncepci „frontend-backend“, což znamená, že je rozdělen do dvou logicky oddělených částí. První část představuje administraci e-shopu (neboli back-end), která je přístupná pouze pro administrátory. A umožňuje obsluhu celého systému – např. vkládání či editaci produktů, objednávek, článků atd. Druhou částí jsou již samotné stránky e-shopu (neboli front-end), které jsou přístupné pro všechny uživatele a zákazníky.

Celý systém se pak rozšiřuje pomocí doplňkových modulů, které na sobě pracují ve většině případů zcela nezávisle. Tato modularita s sebou přináší celou řadu výhod, kdy lze jednotlivé produkty modifikovat na základě potřeb zákazníka. Architektura je tedy založená na spolupráci vzájemně nezávislých služeb (tzv. Service-oriented architecture [1]). A skládá se ze tří vzájemně spolupracujících vrstev – prezentační, aplikační a datové.

3.1 Prezentační vrstva

Prezentační vrstvu představují webové aplikace, které slouží pro komunikaci uživatele se samotným systémem. Jedná se o již zmíněná grafická uživatelská rozhraní – front-end a back-end, jež zajišťují vstupy požadavků a prezentaci výsledků.

U produktu FastCentrik 2.0 se na straně serveru používá technologie ASP.NET MVC, která namísto standardní technologie pro zobrazování UI[6] využívá vlastní řešení – vrstvu založenou na XSL šablonách a jejich transformacích umožňující převod dat do HTML podoby. Na klientské části se pak užívají technologie související s HTML5 a JavaScriptové frameworky jako jQuery, Knockout, Retina atd.

3.2 Aplikační vrstva

Aplikační vrstva, často nazývaná jako Business Logic[7], se využívá převážně k propojení prezentační a datové vrstvy. Spojení s datovou vrstvou je realizováno pomocí ORM. Naopak spojení s prezentační vrstvou je zajištěno díky technologii WCF [2], která k tomuto účelu používá standard webových služeb. Leží zde „jádro“ celé aplikace, její logika, funkce, výpočty a hlavně zpracování dat.

Mimo to se zde nachází i různé doplňující moduly, které pro FastCentrik 2.0 poskytují dodatečnou funkcionalitu – např. cenové hladiny, varianty zboží, slevové kupóny, skladové hospodářství, objemové slevy, hromadné e-maily, napojení na ekonomické systémy a spousty dalších.

3.3 Datová vrstva

Tato vrstva zajišťovala komunikaci s datovým úložištěm (v našem případě Microsoft SQL Serverem) prostřednictvím procedur uložených na straně databázového serveru.

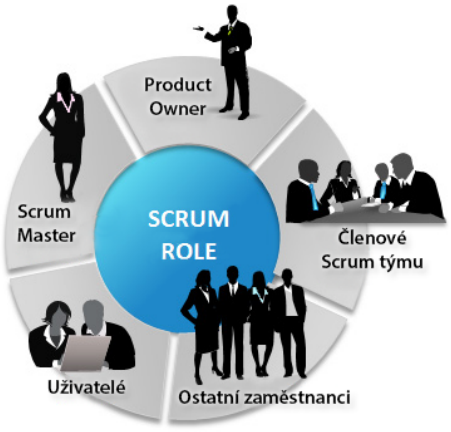
[6] Uživatelské rozhraní
[7] Obchodní logika

4 Metodika vývoje

Již při mém nástupu do NetDirectu jsem byl obeznámen s tím, že celá společnost prosazuje ve svých týmech agilní přístup k vývoji softwaru. Konkrétně šlo o agilní metodiku[8] Scrum. Jejíž klíčovou myšlenkou jsou každodenní setkání celého týmu (tzv. Daily meeting), kde každý člen referuje o své činnosti z předchozího dne, o náplni stávajícího dne a o případných problémech, na které v průběhu narazil.

Metodika prosazuje iterativní vývoj, období iterace se nazývá Sprint a trvá zhruba 1 až 2 týdny v závislosti na domluvě celého týmu (v krajních případech se tato doba může prodloužit až na 4 týdny). Každý Sprint je ukončen tzv. Review meetingem[9], kde jsou předloženy výsledky daného Sprintu (tohoto setkání se mohou zúčastnit kromě členů týmu i lidé z oddělení marketingu, podpory apod.). Průběh tohoto setkání koordinuje tzv. Scrum Master, který kontroluje dodržování určitých pravidel, a zároveň poskytuje případné připomínky společně s tzv. Product Ownerem. Ten může na základě daného jednání jednotlivé úkoly Sprintu akceptovat či naopak vyloučit (popř. sdělit své výhrady). Product Owner s týmem komunikuje i během Sprintu, což předchází zbytečným nedorozuměním vznikajícím zejména na konci těchto Sprintů.

Rozeznávají se zde tedy tyto hlavní role – již zmíněný Product Owner, jehož činností je rozvoj produktu, příprava backlogu[10] produktu, shromažďování priorit a akceptace či upřesňování úkolů pro tým. Dále Scrum Master, který zodpovídá za správné fungování celého vývojového týmu. A Scrum Team Member, což je role všech zbývajících členů vývojového týmu (viz obr. 2).



Obrázek 2: Role v rámci Scrum metodiky

Poznámka 4.1 Daily meetingy se konaly každé ráno a trvaly cca 10-20 minut. Účastníci hovořili jeden po druhém a předávali si slovo pomocí míčku, který si mezi sebou postupně vyměňovali.

[8] Skupina metod pro vývoj softwaru
[9] Přezkumné jednání
[10] Nedokončená práce

5 Správa zdrojového kódu

Tým, který pracuje na vývoji softwaru, pracuje zpravidla s velkou základnou souborů obsahujících zdrojový kód k právě vyvíjenému produktu. Aby jednotliví členové týmu mohli měnit obsah těchto souborů, přidávat nové funkčnosti a opravovat vzniklé problémy, musí se vytvářet různé verze. K tomuto účelu slouží právě verzovací systém TFS neboli Team Foundation Server od Microsoftu. Ten umožňuje sdílet změny s ostatními členy v týmu, a zároveň poskytuje získání jakékoliv dříve uložené verze. Tuto možnost lze využít například při výskytu problému u novější verze. Největším přínosem tohoto CVS[11] je propojení s Visual Studiem, což přináší celou řadu výhod a hlavně určitou míru uživatelského komfortu.

Princip práce s TFS je velice intuitivní, a tudíž se dá pochopit během několika málo dnů. V prvé řadě se musí stáhnout aktuální verze zdrojového kódu na lokální disk, poté již uživatel může kód libovolně modifikovat. Po pečlivém sestavení a otestování modifikovaného kódu se nakonec provede tzv. „Check-in“ neboli nahrání změn zpátky na server. TFS dokonce zvládá do určité míry řešit i konflikty, které vznikají při práci na jednom souboru více uživateli současně. A to tak, že se pokusí dané modifikace sloučit a v opačném případě uživateli nabídne možnost porovnání těchto zdrojových kódů pomocí nástroje „Merge“. Zde si uživatel pouze zvolí, zda chce použít aktuální verzi na serveru anebo svou lokální verzi.

5.1 Podpora metodiky Scrum

Existuje celá řada důvodů, proč vývojové týmy ke své práci využívají verzovací systém TFS. Jedním z nich je také podpora již zmiňované metodiky Scrum. TFS ve Visual Studiu nejenom eviduje veškeré úkoly, ale také zaznamenává různé statistiky a čas, který na nich jednotliví řešitelé stráví. Dokonce TFS od verze 2012 už nabízí i práci s pojmy jako jsou Team, Task Board[12], Product Backlog[13], Sprint Planning a Team Homepage[14], které ještě více zefektivní práci v rámci týmu [3].

Společnost NetDirect k tomuto účelu využívá produkt třetí strany zvaný JIRA[15], který je v určitých ohledech flexibilnější a vhodnější i pro ostatní zaměstnance firmy, kteří se zrovna nepodílí na vývoji produktů. JIRA nabízí řadu nástrojů podporujících projektové řízení, které jsou neustále dostupné přes webové rozhraní. Lze zde sledovat různé aktivity zaměstnanců – zejména komunikaci v rámci týmu, různé reporty, statistiky, stavy projektů atd. Já osobně tento nástroj používal hlavně při řešení chyb na produktu FastCentrik 2.0, kdy jsem pomocí něj evidoval veškerou svou odvedenou činnost.

[11] Systém sloužící ke správě verzí
[12] Nástěnka úkolů
[13] Dodělávky v rámci produktu
[14] Úvodní stránka
[15] Softwarový nástroj pro řízení projektů

6 Řešené úkoly

Náplň své práce bych rozdělil do třech hlavních odvětví. V prvním z nich se zaměřím na práci v rámci servisního oddělení, kde jsem pomáhal při opravách chyb a požadavků od zákazníků. Následně popíšu náplň své práce při vývoji nových funkcionalit pro produkt FastCentrik 2.0 a nakonec se zmíním o mém hlavním úkolu – vývoji webové aplikace nazývané CookBook neboli Kuchařka.

6.1 Testování balíčku

V případě potřeby nasazení změn do produktu FastCentrik 2.0 se na konci Sprintu připravuje tzv. balíček (release[16] produktu do ostrého provozu), který musí být řádně otestován. Mým prvním úkolem bylo tedy otestovat funkčnost celého produktu, který tyto změny zahrnoval. Měl jsem tak vynikající možnost, projít si strukturu FastCentriku a seznámit se s celkovou architekturou projektu.

6.2 Modul ERP Feed List

Jak již samotný název napovídá, jedná se o vytvoření modulu, který v administraci e-shopu zobrazoval seznam URL adres nejpoužívanějších ERP feedů[17]. Implementace modulu do systému nebyla příliš složitá, protože všechny potřebné procedury uložené na straně databáze byly již vytvořeny. Mým úkolem tedy bylo naimplementovat metody, které dané procedury využívaly a zároveň vytvořit komponentu pro zobrazení obsahu na back-endu (viz obr. 3).

	Název	URL adresa	Podrobné logování povoleno
☐	moneyS3/import	http://mustrstagetrunkcz.ms10dev.cz/customdatafeed/1f915cb2-627a-4e01-8a9d-b6e2f65308	×
☐	moneyS3/export	http://mustrstagetrunkcz.ms10dev.cz/customdatafeed/b7d4bb63-9ba1-4913-a04f-a5d2e7cdb	×
☐	vario/import	http://mustrstagetrunkcz.ms10dev.cz/customdatafeed/2403e0cf-d852-4ad2-9fcd-a7d4dd5	×
☐	universalTransport/export	http://mustrstagetrunkcz.ms10dev.cz/customdatafeed/00000000-0000-0000-0000-00006	×

Obrázek 3: Seznam často využívaných ERP feedů

6.2.1 Tvorba modulu

První povinností bylo vytvoření třídy pro daný modul s jedinečným GUID, což je unikátní identifikátor napříč celou aplikací. K tomuto identifikátoru se poté registrují všechny třídy a rozhraní implementující logiku daného modulu. Navíc musí být dodržena správná definice anotací a komentářů pro každý takto nově vytvořený modul.

[16] Uvolnění finální verze
[17] Datový formát poskytující často měněný obsah

Následovalo tedy přidání modulu do back-endu aplikace, kdy se musel vytvořit interface se všemi metodami a proměnnými, které poté byly v back-endu vyvedeny. Ke každé z těchto proměnných se taktéž vztahovala anotace, která nastavovala limity pro hodnotu dané proměnné, viditelnost v rámci administrace a také výchozí hodnotu proměnné. Dále se vytvořil konfigurační soubor, ve kterém byly uloženy názvy všech již nadefinovaných proměnných (pod kterými byly pak v back-endu pojmenovány) a popřípadě i jejich popis (který je v back-endu zobrazen formou uživatelské nápovědy). Nakonec bylo ještě nutné dané rozhraní k již zmíněnému modulu zaregistrovat.

Aby byly metody definované v rozhraní přístupné v administraci, musel jsem je přidat také do kontraktů pro WCF službu. Poté jsem veškeré metody řádně okomentoval a nakonec pro modul napsal unit testy.

6.2.2 Zprovoznění v rámci aplikace

Celou práci jsem pak nahrál na server pomocí již zmíněného verzovacího systému TFS. A aby se změny provedené v rámci této solution[18] projevily v celé aplikaci, musely se znovu sestavit všechny související solutions. Zde se muselo dbát na pořadí těchto Re-buildů[19], kdy se vždy začínalo sdíleným solutionem, pak následovaly „dvojice“ solutionů pro Core[20], ShopCentrik a MediaCentrik. A končilo se znovusestavením dvou nejčastěji modifikovaných řešení nazývaných jako Host a Admin. Důvodem tohoto počínání nebyl pouze rozsah celkového řešení (který musel být kvůli své velikosti rozdělen na menší části), ale i zachování modularity a hlavně nahraditelnosti jednotlivých částí. Ty se pak jednodušeji kompilovaly do samostatných DLL knihoven. Pokud jsme potřebovali nějaký modul otestovat i na ostrém řešení, musely se vybrané projekty zastavit přes aplikaci MS10 Project manager. Zde se poté spustil Deploy[21] a nakonec se zastavené projekty opět spustily.

Poznámka 6.1 Tento modul sloužil hlavně pro interní účely firmy. Pracovala s ním ve velké míře zákaznická podpora a hlavně telemarketeři, kteří pomáhali zákazníkům s nastavením propojení FastCentriku 2.0 s podporovanými ERP systémy.

6.3 Úprava regex kódů

Dalším přiřazeným úkolem byla úprava poštovních regex kódů uvnitř celé stávající aplikace. Ty se používaly například pro validaci vstupu adres uživatelů v rámci objednávkového procesu. Uživatel měl možnost vybrat si u jednokrokové či vícekrokové objednávky svou zemi. Vzhledem k tomuto výběru se poté prováděla validace poštovního směrovacího čísla (neboli PSČ). A právě k tomuto účelu sloužily regex kódy, které reprezentovaly regulární výrazy, na jejichž základě se prováděla daná validace. Jednalo se o řetězec

[18] Struktura pro organizaci projektů
[19] Znovusestavení
[20] Celkové jádro systému
[21] Nasazení releasu

znaků skládající se z literálů textu, které se měly shodovat, a speciálních znaků, které nebyly součástí hledaného textu, popisujících celou množinu alternativ, jež tyto požadavky splňovaly.

Náplní tohoto úkolu byla tedy úprava stávajících poštovních regex kódů pro zhruba 216 zemí, přičemž jsem se měl primárně zaměřit na evropské státy a zahrnout zde i souhrnný regex kód pro všech 50 států USA. Níže jsem pro názornost uvedl ukázku poštovního regex kódu pro Velkou Británii.

```
<Country Code="GB" Number="826" Name="United Kingdom">
    <PostalCodeRegex>
        ^(GIR 0AA)|([A-PR-UWYZ]((\d(\d|[A-HJKSTUW])?)|([A-HK-Y]\d
        (\d|[ABEHMNPRV-Y])?)) \d[ABD-HJLNP-UW-Z]{2})$
    </PostalCodeRegex>
</Country>
```

6.4 Rozšíření o Phone Regex

Na základě úspěchu s úpravou poštovních regex kódů jsem byl pověřen o doplnění regulárních výrazů i pro telefonní čísla. Ty se odvíjely taktéž na základě předchozího výběru země uživatelem (respektive zadáním země v jeho doručovací adrese). Zde jsem narozdíl od poštovních regexů implementoval pouze evropské státy a velmoci jako je Rusko či USA. Níže opět uvádím názornou ukázku snadněji čitelného regex kódu. Tentokrát se jedná o regulární výraz pro kontrolu vstupu u telefonních čísel pro uživatele s doručovací adresou v České republice.

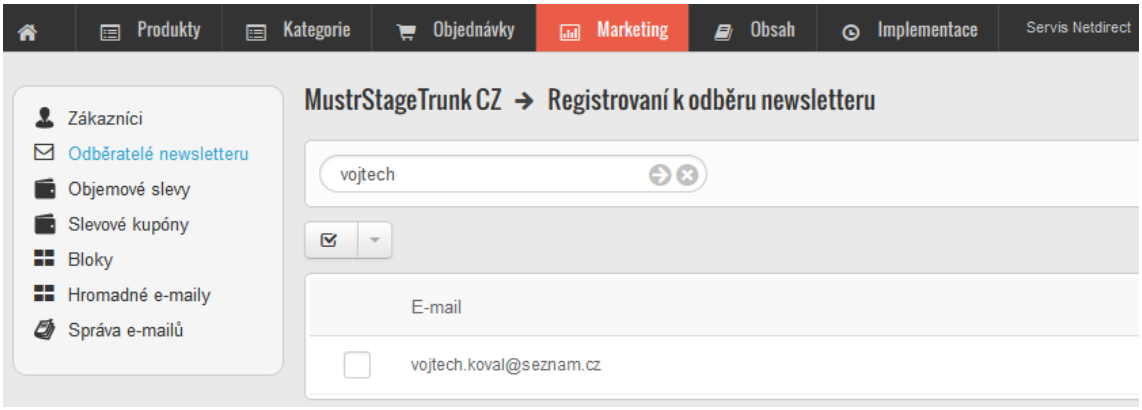
```
<Country Code="CZ" Number="203" Name="Czech Republic">
    <PhoneRegex>
        ^((\+420|00420)[ ]?)?\d{3} ?\d{3} ?\d{3}$
    </PhoneRegex>
    <PhoneCountryCode>420</PhoneCountryCode>
</Country>
```

Po nasazení této funkčnosti do ostré verze jsem se ovšem setkal s určitými problémy, kdy chtěl uživatel například zadat své telefonní číslo s jinou předvolbou, než podporuje jeho země. Typickým příkladem je, že uživatel s doručovací adresou uvedenou v České republice vlastní slovenskou SIM kartu, a sice má předvolbu „+421“. Ta už ovšem není pro Českou republiku podporována, a tudíž validace bude hlásit špatně zadaný formát. Proto se tato funkčnost musela částečně předělat tak, aby k podobným situacím již nedocházelo.

Poznámka 6.2 Tato zkušenost mě přesvědčila o tom, že ačkoliv je aplikace napsána co možná nejlépe a je taktéž důkladně otestována, občas tyto skutečnosti nestačí a musí se myslet i na speciální případy, které mohou nastat do budoucna.

6.5 Filtr odběratelů newsletteru

V administrační části webové aplikace se nachází seznam všech odběratelů newsletteru. Jedná se o uživatele, kterým jsou pravidelně zasílány e-maily obsahující novinky v rámci marketingových kampaní. Mým úkolem bylo upravit stávající filtr tak, aby umožňoval filtrovat uživatele nejenom na základě jména či příjmení, ale také i podle jejich e-mailové adresy (viz obr. 4).



Obrázek 4: Ukázka vyhledávání odběratelů newsletteru

Našel jsem si tedy všechny metody, které vracely na základě parametrů filtru seznam zákazníků. Poté jsem si dohledal jejich namapování na procedury uložené na straně serveru a vytvořil si vlastní proceduru, která vracela uživatele na základě e-mailu. Nicméně vzápětí jsem narazil na problém, kdy filtr vracel neúplné výsledky. A tudíž jsem se rozhodl předělat vyhledávání filtru na fulltextové, kdy se data serializovala pomocí procedury do XML souboru. Tento princip pracoval s vyhledávanými daty mnohem lépe, než předchozí řešení. A modifikace se poté uplatnila i na jiných místech produktu, kde bylo použito vyhledávání právě pomocí filtru.

Mé řešení tedy zahrnovalo úpravu stávajících metod v repozitáři newsletteru a vytvoření zcela nové procedury, ke které jsem si poté nechal vygenerovat její namapování. Ke generování jsem využil firemní nástroj nazývaný SprocGenerator – ten zajišťoval namapování všech procedur, uložených na straně databáze, používaných v rámci celé aplikace. Níže uvádím příklad metody, která s daným filtrem pracovala.

```
public NewsletterRegister GetNewsletterRegister(NewsletterRegisterFilter p_filter )
        {
            Sproc.NewsletterReg_Get proc = new Sproc.NewsletterReg_Get(this.Database);
            proc.pkTblCOR_NewsletterRegistred = p_filter.NewsletterRegisterId;
            proc.sEmail =  p_filter .Email;
            using (IDataReader reader = proc.ExecuteReader())
                    {
                            return new NewsletterRegisterMapper().Map(reader);
                    }
        }
```

Výpis 1: Ukázka metody *GetNewsletterRegister*

6.6 Modifikace šablon

Obsahem dalšího úkolu byla úprava stávajících HTML šablon, které se zákazníkům zasílaly formou e-mailů při registraci a vytvoření objednávky zboží na e-shopu. Tato modifikace se týkala anglické verze FastCentriku 2.0, kdy bylo potřeba zajistit překlady českých názvů a nahradit některé HTML elementy. Nejsložitějším úkonem zde bylo zjistit, kde se dané šablony nacházejí. K tomu jsem využil debugger[22], pomocí kterého jsem zjistil, které metody se při odeslání těchto e-mailů volají. Na tomto základě jsem pak odhalil, s jakými uloženými procedurami spolupracují. Zde jsem si také vyzkoušel práci s nástrojem SQL Server Profiler, který zobrazuje trace[23] všech databázových operací, pomocí kterého se dají dohledat aktuálně používané procedury uložené na straně databáze. Bylo zde ovšem nutné pracovat na ostrém projektu, protože lokální verze odesílat e-maily neumožňovala.

Poté jsem si pomocí SQL Server Management Studia[24] vypátral dané uložené procedury v databázi a zjistil, s jakými daty pracují. Tyto data jsem si poté taktéž dohledal a zjistil, že šablony byly v databázi uloženy pomocí serializace. Navíc byly i komprimovány, aby se redukovala jejich velikost. Mým úkolem bylo tedy rozparsovat tyto serializované XML data. K tomuto účelu jsem využil Visual Studio, které umí XML soubory přehledně formátovat. Poté přišla na řadu fáze, kdy jsem si musel z body elementů šablon vykopírovat obsah sekce zvané CDATA. Ta slouží pro uložení dat, u kterých chceme potlačit interpretaci jejich značení. Používá se hlavně pro uložení zdrojových kódů, kde se často pracuje se speciálními znaky (k těm se vyjádřím v následujícím úkolu). Výše zmíněný obsah jsem poté modifikoval dle požadavků, dokud jsem s výsledkem své práce nebyl spokojený.

Následovala opětovná komprimace celého XML kódu do původní podoby, k čemuž jsem použil vlastní aplikaci. Ta obsahovala algoritmus, který vynechal veškeré nepotřebné „bílé“ znaky (jako jsou mezery, tabulátory a znaky pro zalomení řádků). To všechno jsem nakonec předal databázistům k nasazení na ostrou verzi, což už nebylo v mé kompetenci (kvůli zachování bezpečnosti na ostrých databázích).

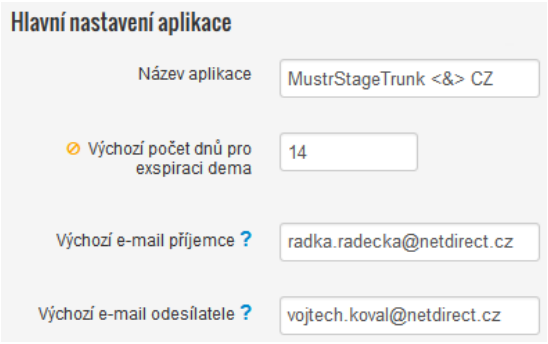
Dále jsem musel upravit resources[25] v „.resx“ souborech, které se skládaly ze záznamů ve formátu XML specifikujících objekty a řetězce uvnitř těchto XML tagů. Ty se využívaly zejména pro lokalizaci aplikace do cizích jazyků, kdy se na jejich základě posílala data do překládací aplikace, ve které se již konkrétní překlad do anglické, polské či bulharské verze zařídil. K této aplikaci jsem ovšem neměl přístup – o její správnou funkčnost se starali zkušenější kolegové.

[22] Nástroj pro hledání chyb ve fázi ladění
[23] Výpis trasy
[24] Prostředí pro správu SQL Serveru
[25] Zdroje

6.7 Parsování názvů e-shopů

Jak jsem již v úvodu této práce zmiňoval, v průběhu praxe jsem byl nějakou dobu součástí servisního oddělení. To mělo za úkol udržovat správný chod všech již nasazených e-shopů, a tudíž řešit veškeré vzniklé problémy. Určitá část problémů se bohužel odhalila až samotnými zákazníky, kteří se poté obraceli na zákaznické centrum NetDirectu. Zde se daný problém popsal a zaslal na servis ve formě tzv. „ticketu“[26]. Ten se poté objevil ve webovém rozhraní již výše popsaného nástroje pro řízení projektů JIRA. Kde se nacházela fronta takto nevyřešených problémů, ze které jsem si vždy jeden ticket vybral a přiřadil se k němu jako řešitel.

Jedním z těchto problémů byl případ, kdy zákazníkovi nešel zobrazit front-end jeho e-shopu. Zkontroloval jsem tedy jeho nastavení v administraci obchodu. Nicméně se vše zdálo být v pořádku, a tudíž jsem si nechal stáhnout zálohu jeho ostré databáze k sobě na lokální disk. Odtud jsem jej mohl dále ladit pomocí debuggeru a přijít tak na chybu, která se v daném případě vyskytla. Po delším bádání jsem zjistil, že se chyba týká parsování jedno z XML souborů. Tudíž jsem jej analyzoval, načež jsem vzápětí odhalil příčinu této chyby. Byly jím speciální znaky vyskytující se v názvu e-shopu (konkrétně se jednalo o znak „&“). S těmito znaky se musí v XML od verze 1.0 zacházet jako s bílými znaky, a proto jsem je musel nahradit za escape sekvence[27]. Což znamenalo změnit způsob ukládání těchto znaků při tvorbě jakéhokoliv XML souboru obsahujícího právě název e-shopu.



Obrázek 5: Nastavení názvu e-shopu

Tato změna se tudíž týkala všech již nasazených e-shopů a hlavně těch, které by do budoucna v názvu obsahovaly některý z těchto speciálních znaků. Musel jsem tedy vyhledat veškeré komponenty aplikační vrstvy, které s názvem obchodu pracovaly, a na jejich základě nalézt nejlepší způsob, jak daný problém vyřešit. Nakonec jsem odhalil jednu z obslužných metod, která se využívala právě pro ukládání XML šablon do databáze. Zde jsem tedy provedl opatření, aby se všechny stávající speciální znaky v názvu e-shopu nahradily kódy z ASCII tabulky.

Po této implementaci se daná funkčnost musela vyzkoušet, aby se předešlo možným problémům při nasazení na ostré projekty. Pro tento účel se využila bulharská verze ostrého e-shopu, se kterou v danou chvíli nikdo nepracoval. Zde se dané změny aplikovaly a provedla se i úprava názvu (viz obr. 4) e-shopu tak, aby obsahoval již zmíněné bílé znaky. Následně jsem důkladně otestoval veškeré moduly, které s názvem obchodu pracovaly. A až po tomto kroku se tato úprava zahrnula do další verze balíčku, který se poté na konci Sprintu nasadil všem provozovatelům e-shopů.

[26] Úkol k vyřešení
[27] Kódování řetězcových literálů

6.8 Další řešené tickety

Kromě výše popsaných úkolů jsem se v rámci praxe zabýval řešením spousty dalších ticketů, kdy se jednalo především o různé menší změny či úpravy. Ty bych se pokusil v následujících odstavcích pouze lehce nastínit a pokud možno doplnit o způsoby jejich řešení. Hned v úvodu bych chtěl také upozornit na fakt, že ne všechny úkoly bylo možno vyřešit – ať už kvůli nemožnosti daný problém reálně nasimulovat či kvůli časové tísni.

6.8.1 Ekonomický systém Pohoda

Jedním z těchto úkolů bylo opravit chybu, která se objevovala v rámci logovacího souboru z účetního programu Pohoda. Zde jsem zjistil, že se jedná o problém s koncovým datem u importního modulu. Tento problém jsem však nemohl dokončit, protože mi k danému produktu nebyla poskytnuta licence. Celý tým totiž vlastnil pouze omezený počet těchto licencí. Ty byly NetDirectu uděleny na základě partnerství se společností Stormware, která daný produkt nabízela. Následně jsem tedy tento problém (lépe řečeno ticket) přiřadil na kolegu, který již danou licenci vlastnil.

6.8.2 Úpravy na míru zákazníkovi

Dalším řešeným ticketem byla úprava front-endu zákazníkova e-shopu na základě jeho předešlé prosby. Zde se jednalo o modifikaci kaskádových stylů v externím CSS souboru, kdy jsem pro dosažení konečného vzhledu webových stránek využil nových selektorů verze 3.0 – úprava zahrnovala modifikaci menu, seznamu produktů a patičky e-shopu. Následně jsem se ještě zabýval problémem, kdy si tento zákazník stěžoval, že vygenerovaný HTML kód jeho front-endu obsahuje URL adresu odkazující do administrace obchodu. Zde šlo o nesprávný způsob uložení některých obrázků ze strany zákazníka, kdy se namísto relativních adres obrázků vypisovaly adresy absolutní. Musel jsem tedy změnit umístění všech takto uložených obrázků.

6.8.3 Modul Ulozenka.cz

Tento ticket zahrnoval opravu problému, kdy se při přenosu dat přes FTP do systému Ulozenka.cz nepřepisovaly telefonní čísla. Služba Ulozenka.cz je provozována společností Uloženka s.r.o. a slouží k výdeji zásilek na předem určených výdejních místech. Jednoduše řečeno, když si zákazník vybere jako formu doručení jedno z těchto výdejních míst, systém automaticky přepošle službě Ulozenka.cz XML soubor, který obsahuje základní údaje zákazníka – jako jméno, příjmení, telefonní číslo atd. (kromě jeho adresy, která by v daném případě postrádala smysl). Šlo tedy o korekci exportního modulu, ve kterém se přenášela tato data do XML formátu.

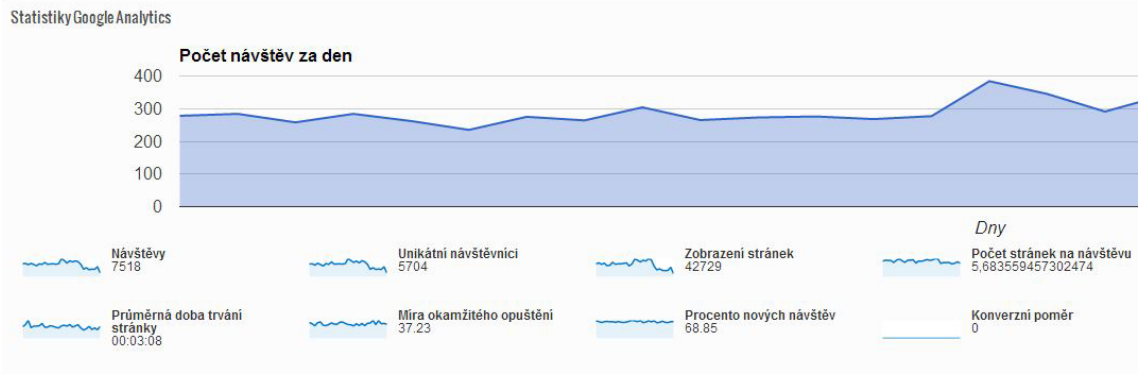
6.8.4 Šablona nového produktu

Některé tickety, které jsem postupem času řešil, byly označeny jako kritické. Mezi ně právě patřil i následující problém, který se týkal nemožnosti založení nového produktu.

Zákazníci si ihned po vydání nového balíčku začali stěžovat na tuto chybu, kdy v administraci nemohli vytvořit jakýkoliv nový produkt. Musel jsem tedy na tento problém urychleně reagovat, a proto jsem okamžitě začal zkoumat příčiny vzniku této chyby. V aplikační vrstvě se vše zdálo být v pořádku a ani debugger žádnou chybu nehlásil. Musel jsem se tedy zaměřit více do hloubky. Po asi hodině usilovného zkoumání jsem si na pomoc přizval kolegu, s jehož pomocí jsme nakonec daný problém vyřešili. Jednalo se o špatně napsaný JavaScriptový kód využívající JS knihovny zvané Knockout, kdy chyběl tzv. data-binding[28] mezi některými objekty.

6.8.5 Statistiky Google Analytics

Jedním ze základních modulů dostupných v rámci produktu FastCentrik 2.0 je modul Google Analytics, který slouží pro zobrazování statistik služby Google Analytics přímo na úvodní stránce back-endu. Služba je dostupná zcela zdarma. Jedinou podmínkou pro její aktivaci je registrace e-shopu na oficiálních webových stránkách společnosti Google `http://www.google.com/intl/cs_ALL/analytics`. Dále již stačí pro její aktivaci pouze vložit vygenerované ID účtu do administrace obchodu. Posléze zákazník na oplátku získá optimální přehled o provozu jeho e-shopu a efektivitě marketingu přímo v administraci jeho obchodu (viz obr. 6).



Obrázek 6: Graf návštěvnosti e-shopu

Ačkoliv v tomto modulu nebyly provedeny delší dobu žádné zásahy, začaly se na něj hromadit ze strany zákazníků stížnosti. Ve většině případů šlo o problémy s jeho zobrazením. Mou povinností tedy bylo daný problém nasimulovat a pokud možno co nejrychleji vyřešit. Zde jsem ovšem narazil na situaci, kdy ani po různých úpravách nastavení této služby nešlo dané chování nasimulovat. Po delším zkoumání jsem vydedukoval, že se tento problém vyskytoval hlavně u nově příchozích provozovatelů e-shopů. U stávajících zákazníků se toto chování neobjevovalo. Problém tedy nebyl ve špatném nastavení služby Google Analytics, nýbrž přímo ve službě samotné. Musel jsem tedy detailněji prozkoumat modul, který se službou komunikoval.

[28] Datová vazba

Nakonec jsem vypátral, že modul využíval již zastaralou verzi přihlašování k této službě – novější verze totiž požadují opisování tzv. CAPTCHA kódu[29]. Jednalo se tedy o vývoj zcela nové funkcionality, která se musela nejdříve odsouhlasit vedením vývoje. A proto jsem s danou situací seznámil Product Ownera, který se již postaral o začlenění vývoje dané funkcionality do následujícího Sprintu.

6.8.6 Nedokončené úkoly

Jak jsem již v úvodu této kapitoly zmiňoval, některé úkoly nebylo možno vyřešit – ať už kvůli nemožnosti daný problém reálně nasimulovat či kvůli časové tísni. Většina z nich byla tedy buď zrušena, nebo přeorganizována (či převedena do jiného týmu). Níže bych uvedl seznam obdobných úkolů a zahrnul zde i některé méně významné či dokonce „triviální“ úlohy:

- Triviální úloha, kdy se zákazníkovi načítal do hlavičky webových stránek obrázek, který nebyl viditelný, a tudíž zpomaloval načítání celého e-shopu.
- Ticket, kdy si uživatel mohl do košíku přidat více množství, než měl e-shop skladem.
- Modifikace JS kódů pro dosažení očekávaných výsledků (např. upravení input elementů u formulářů).
- Rozsáhlý problém, kdy uživateli nešla načíst administrace e-shopu kvůli velkému množství dat.
- Úprava CSS stylů, kdy se muselo sjednotit zobrazení webových stránek v různých prohlížečích.
- Problém, kdy byly u modulu „Hlídací pes“ zasílány registrovaným zákazníkům e-maily o zlevnění aktuální ceny, ačkoliv se cena nijak nezměnila (nebyly implementovány ceníky zákazníků – každý ceník může mít jinou globální slevu).

6.9 Vývoj webové aplikace

Jednoznačně největší časový úsek celé praxe představovala práce na vývoji nové webové aplikace nazývané CookBook či Kuchařka pro ThemeCentrik. Aplikace ThemeCentrik primárně slouží pro získání originálního grafického vzhledu, který se dá posléze aplikovat na jakémkoliv e-shopu postaveném na systému FastCentrik 2.0. Jedná se tedy o kolekci již předem připravených šablon, ze kterých si mohou zákazníci vybírat. Celý seznam těchto šablon je dostupný na následující adrese: `http://themes.netdirect.cz`. Zde mají zákazníci na výběr buďto méně originální bezplatné šablony, nebo atraktivnější placené šablony.

Po objednání jedné z těchto variant se zákazníkovi aplikuje vybraný motiv na celý jeho e-shop. Velmi často se ale stává, že zákazník strukturu svého e-shopu chce následně

[29] Turingův test k odlišení uživatelů od robotů

upravit dle svých požadavků. Zde ovšem naráží na problém, kdy je nutné dodržovat určitá pravidla spojená s modifikací tohoto produktu. A právě pro tento účel byla určena již zmíněná webová aplikace CookBook. Ta se skládala ze dvou hlavních částí:

- **Struktury** – popis celého systému včetně layoutů, šablon a seznamu všech povinných či volitelných komponent.
- **Implementace** – soupis všech pravidel a doporučení, které musí partneři a zejména designéři (či grafická studia) při tvorbě či modifikaci šablon striktně dodržovat (popřípadě je alespoň znát a řídit se jimi).

6.9.1 Obecně o aplikaci

Jak jsem již uvedl výše, aplikace by měla sloužit zákazníkům jako tzv. „manuál“ k produktu FastCentrik 2.0. Mělo by se tedy jednat o dynamickou webovou prezentaci obsahující aktivní prvky a data, která bude zobrazovat a načítat z externího zdroje (v našem případě z databáze). Tento způsob prezentace by tak umožňoval data centralizovat, čímž by se napomáhalo k efektivnějšímu spravování obsahu stránek. Mým podkladem pro samotnou implementaci byly nepřehledné statické stránky psané ve skriptovacím jazyce PHP, které obsahovaly pro každou stránku samostatný PHP soubor. Na jejich základě jsem následně vytvářel již zmíněnou dynamickou webovou prezentaci, která byla pro takové množství stran mnohem vhodnějším řešením.

6.9.2 Zvolený postup řešení

Základem tvorby zcela nové aplikace bylo určení jednotlivých fází vývoje (či lépe řečeno etap), podle kterých jsem poté „krok za krokem“ postupoval. Níže uvádím obecný harmonogram (neboli časově seřazený seznam) těchto kroků:

1. Návrh datové struktury
   - Tvorba konceptuálního modelu (ERM)
   - Tvorba logického (relačního) modelu
2. Vytvoření databáze a naplnění daty
3. Nasazení webového serveru přes IIS[30]
4. Založení MVC aplikace
   - Implementace datové struktury – Modelu
   - Tvorba základního Controlleru
   - Postupná implementace View (pomocí XSLT šablon)
5. Testování funkčnosti
   - Nasazení na ostrý server

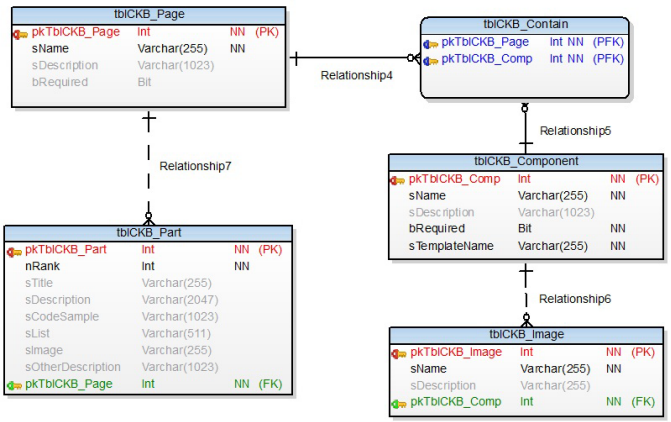
[30] Správce Internetové informační služby

6.9.3 Rozvržení aplikace

Základním kamenem celé aplikace bylo správné navrhnutí datové struktury, která do značné míry ovlivňovala bezporuchovost, udržovatelnost a hlavně rozšiřitelnost výsledné aplikace. Při datovém modelování je nutné nejprve vytvořit konceptuální datový model, který představuje určité zobecnění oproti konkrétní implementaci datové struktury v relační či objektové databázi (nezávislost modelu na konkrétním databázovém systému). Konceptuální model (dále pouze ERM) jsem vytvořil pomocí nástroje nazývaného Toad Data Modeler, který umožňoval modelovat databázové struktury pomocí vizuálních prvků. Zde jsem musel dávat pozor na dodržování firemních zvyklostí ohledně pojmenování jednotlivých sloupců, tabulek atd. Tento nástroj jsem poté taktéž použil na převod daného ERM na model relační (neboli logický), který sdružoval data do tzv. „relací“ neboli tabulek tvořících základ relační databáze.

Dále jsem si pomocí tohoto modelovacího nástroje nechal vygenerovat SQL skripty, pomocí kterých jsem si následně vytvořil vlastní databázi v SQL Server Management Studiu. Zde jsem poté založil vlastní namespace[31] pro databázové schéma Kuchařky nazvaný „ckb“, který jsem následně nastavil u všech aktuálně vygenerovaných tabulek (namísto výchozího schéma „dbo“). A také jsem opravil zbývající nedostatky jako např. pojmenování jednotlivých vztahů mezi relacemi.

Tento proces jsem musel v průběhu vývoje několikrát opakovat z důvodů různých modifikací, kdy bylo nutné změnit stávající datovou strukturu. Například při přidání nové tabulky, která uchovávala další specifická data (konkrétně třeba při přidání relace tblCKB_Image pro uchovávání záznamů o obrázcích). Pro přiblížení dané struktury jsem zde uvedl vizuálně zpracovaný diagram jedné z verzí relačního modelu (viz obr. 7). Nicméně tato verze byla později modifikována z důvodu přidání lokalizace do jiných jazyků (především angličtiny), o které se zmíním později.



Obrázek 7: Ukázka relačního modelu

Po úspěšném vytvoření databázové struktury bylo ovšem nutné naplnit prázdnou databázi daty. K tomuto účelu jsem si pro každou entitu vytvořil vlastní T-SQL skript skládající se z INSERT příkazů, které poté tyto data záznam po záznamu vkládaly. Zde bylo taktéž nutné dávat pozor na inkrementace[32] hodnot primárních klíčů, které se musely předem nastavit.

[31] Jmenný prostor
[32] Navyšování hodnoty

Konkrétní data jsem pak čerpal z již zmíněné předlohy, která obsahovala spousty gramatických chyb. Musel jsem tedy důkladně kontrolovat formulace jednotlivých slovních spojení, případně je nahrazovat smysluplnějšími výrazy (nezbytnou součástí byla korekce interpunkce a také doplnění chybějících URL adres, které odkazovaly na další webové podklady).

6.9.4 Implementace aplikace

Následně přišla na řadu samotná implementace aplikace. Tu jsem realizoval ve vývojovém prostředí .NET [4], programovacím jazyce C# [5] a jako architektonický vzor mi sloužil Model View Controller (dále pouze MVC). Tento návrhový vzor rozděluje datový model aplikace, uživatelské rozhraní a řídící logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní. A jak již samotný název této architektury napovídá, těmito komponentami jsou Model (doménová vrstva), View (již zmíněné uživatelské rozhraní) a Controller, který reaguje na události a zajišťuje změny jak v Modelu, tak i ve View. MVC se tedy hodí především pro složitější webové aplikace, kde zajišťuje flexibilitu a spolehlivost. Díky čemuž se pro vývoj této aplikace zdál být ideální volbou.

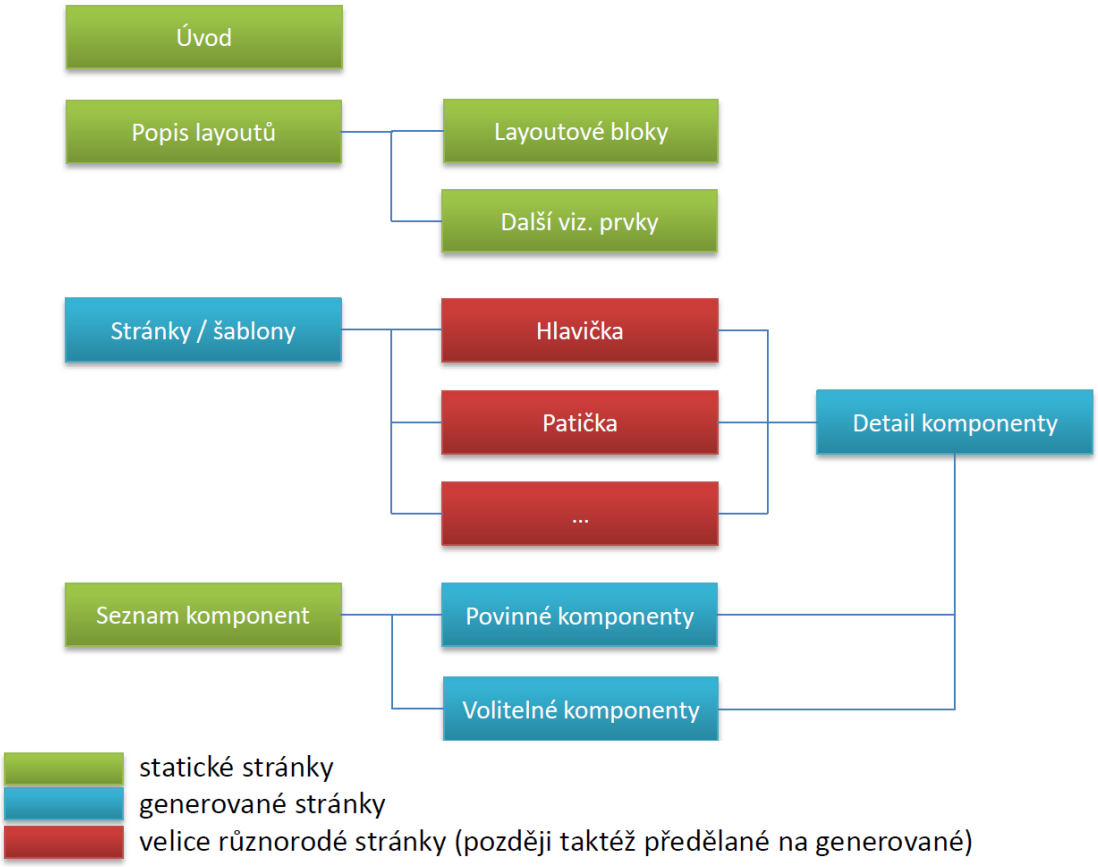
Ještě před založením nového projektu mi byla poskytnuta již předpřipravená šablona (rozpracovaný projekt), díky které jsem měl možnost zorientovat se v už zaběhnutých technologiích v rámci vývoje. V prvé řadě jsem si uvědomil, že se nejednalo o klasickou aplikaci využívající ASP.NET MVC [6]. View totiž nevyužívalo standardní view engine Razor[33], který je součástí již zmíněného MVC už od verze 3. Namísto toho byla použita vlastní vrstva založená na přenosu informací (lépe řečeno dat) pomocí XML a na XSL šablonách, které dané data převáděly prostřednictvím XSL transformací (dále jen XSLT [7]) do HTML formátu.

V prvé řadě bylo nutné přidat reference na firemní knihovny, které se staraly o určitou část funkčnosti (např. o zmíněný převod XSL šablon do HTML kódu). Dále se musely zaregistrovat globální filtry a tzv. „routes“[34], které umožňovaly využívat URL adresy bez nutnosti mapování ke kontrétním souborům webové aplikace. Následně jsem začal vytvářet doménovou vrstvu, která obsahovala veškerou Business logiku (zkráceně BL). Ihned nato jsem se pustil do tvorby datové vrstvy, kde jsem pro namapování jednotlivých databázových tabulek na třídy využil návrhový vzor Repository (česky nazývaný repozitář). Ten zajišťoval, že Model aplikace netušil, jaká třída se starala o práci s daty (Model by se tedy nezaobíral situací, kdy by se daná třída v budoucnosti nahradila jinou třídou s odlišnou funkcionalitou). Tento vzor taktéž umožňoval snadné provádění unit testů, což bylo značnou výhodou.

Na následujícím příkladě bych chtěl nastínit část struktury aplikace pro lepší pochopení dané problematiky – uvedl jsem zde stručný přehled jednotlivých stránek (viz obr. 8), ve kterém jsou stránky rozděleny do tří skupin (statické, generované a různorodé).

[33] Syntaxe používaná pro tvorbu dynamických webů
[34] Trasy

Statické stránky měly vytvořené své vlastní unikátní View, kdežto generické stránky využívaly XSL šablony, do kterých se postupně plnila data získaná z databáze prostřednictvím uložených procedur. Tyto šablony se poté transformací převedly do HTML formátu, který se záhy publikoval ve webovém prostředí. A aby toto bylo možné, musel se ještě naimplementovat Controller, který obsahoval metody s návratovou hodnotou ActionResult, která se starala právě o vrácení aktuálně požadovaného View (popřípadě o přesměrování na další akci či předání obsahu bez vlastního View).



Obrázek 8: Základní struktura první části webové aplikace

Poznámka 6.3 Pojem „velice různorodé stránky“ zahrnuje všechny stránky s odlišnou strukturou. Tudíž se jednalo o vizuálně odlišné stránky, které se však skládaly z obdobných prvků (jako jsou nadpisy, odstavce, obrázky, seznamy, ukázky zdrojových kódů atd.). Musela se tedy vytvořit velice spletitá XSL šablona, která se o zobrazení všech takovýchto stránek starala. Zde bylo velmi náročné zachovat původní rozpoložení prvků na stránce (dle původní šablony).

Kromě již zmíněných povinností bylo ještě nutné v aplikaci nakonfigurovat tzv. „connection string“, který sloužil pro přístup k SQL databázi. A aby bylo možné aplikaci ladit pomocí debug režimu, musela se nasadit na skutečný IIS server. K tomuto účelu

jsem využil Správce Internetové informační služby ve verzi 6.0, který musel být nejdříve nainstalován (popřípadě musela být povolena jeho funkce v samotném operačním systému, která je defaultně vypnuta). Následně již nic nebránilo postupnému vývoji nových funkcionalit této aplikace.

6.9.5 Modifikace požadavků

V rámci své práce jsem se neustále setkával s různými změnami požadavků, které na mě byly v průběhu praxe kladeny. Což pro mě bylo z počátku velice nezvyklé. Nicméně po určité době jsem si na tyto změny postupně navykl a v současné době již vím, že se bez nich vývoj softwaru nedokáže obejít. Osobně tento nezvyk beru jako velice přínosnou zkušenost, o kterou jsem byl do té doby ochuzen. Neustálé vyvíjení nového úsilí nutí člověka hodně přemýšlet, což vede k efektivnějšímu způsobu myšlení a taky k lepšímu pracovnímu nasazení.

S prvními změnami jsem se setkal již při počátečních návrzích datové struktury, kdy bylo nutné provést několik desítek různých modifikací, což vedlo ke vzniku různých verzí (které jsou neodmyslitelnou součástí vývoje). Musel jsem si taktéž uvědomit, že ne vždy je funkční aplikace požadovaným výstupem práce. Často jsem se setkával s případy, kdy se musela část kódu kompletně předělat. Dokonce jsem se setkal i se situací, kdy se kvůli neefektivitě dané funkčnosti měnila i celá technologie. Což v jisté míře změnilo můj náhled na vývoj softwaru a obohatilo mé dosavadní zkušenosti.

Takovým případem byl např. požadavek, kdy jsem měl již funkční aplikaci lokalizovat do jiného jazyka (konkrétně angličtiny) a zároveň umožnit i její rozšiřitelnost do jakéhokoliv dalšího překladu. Musel jsem tedy změnit celou datovou strukturu včetně různých vazeb a výchozích pojmenování (např. u obrázků, které byly nastaveny implicitně). Vše muselo rázem splňovat určitou modularitu. Musely se tedy založit další relace pro zajištění podpory ostatních jazyků. To vedlo k celkové změně všech dosavadních procedur uložených na straně databáze a modifikaci celé aplikační vrstvy včetně způsobů mapování a cachování získavaných dat. Dále bylo nutné zavedení tzv. „resources“[35], které dokázaly reagovat pomocí ResourceManagera na změnu kultury prostředí. Tyto datové zdroje se ovšem využívaly pouze pro překlady slovních spojení, které nebyly obsaženy v samotné databázi (šlo tedy o naprosto mizivou menšinu slov). A starala se o ně již popisovaná překládací aplikace. Všechny tyto funkčnosti poté reagovaly na změnu kultury uživatelského prostředí. Kultura se následně ukládala a dohledávala v tzv. „cookies“[36], pomocí kterých si prohlížeč po určitou dobu pamatoval poslední známé nastavení. Kvůli tohoto řešení se musel upravit i způsob skládání URL adres přes routes, kdy jsem musel zcela zamezit výskytu otazníku v adrese. Ten se totiž používal pro nastavení parametrů lang (např. `?lang=en`) a debug (např. `?debug=on`).

[35] Soubor datových zdrojů
[36] Data uložená na straně prohlížeče

6.9.6 Refaktoring kódu

Následně jsem se také věnoval refaktoringu[37] celé aplikace, kdy jsem upravoval jak uložené procedury, tak i samotnou business logiku. Především šlo o zredukování počtu operací prováděných na straně databázového serveru. Dále jsem se snažil upravit i způsob cachování dat a jejich procházení pomocí LINQ[38] dotazů. Kromě toho jsem sjednotil formát veškerých obrázků a změnil způsob načínání všech externích souborů (skriptů, kaskádových stylů, obrázků atd.), kdy se zamezilo používání absolutních adres. Následně jsem optimalizoval a zkomprimoval všechny CSS soubory pomocí nástroje CSS Minifier dostupného na adrese `http://cssminifier.com`, což vedlo ke zkrácení doby načítání webových stránek. Navíc jsem zavedl i tzv. „CSS Image Sprites“[39], které zredukovaly počet požadavků na server, což také významně napomohlo ke zkrácení této doby. Všechny tyto změny následně vedly ke zrychlení běhu celé aplikace.

6.9.7 Unit testování

Každý vývojář se při vývoji softwaru potýká s výskytem chyb různého charakteru. Pro jejich eliminaci je nutná tvorba tzv. unit testů, jejichž cílem je ověřování správné funkčnosti dílčích částí zdrojového kódu. Ačkoliv je tyto testy lepší psát před samotnou implementací aplikace, já se jim věnoval až v případech, kdy bylo nutné otestovat nějakou funkčnost bez nutnosti neustálého spouštění aplikace. Například v situaci, kdy jsem chtěl zkontrolovat data načtená z databáze. Nebylo tedy nutné inicializovat při každém ladění celou aplikaci odznovu, nýbrž stačilo otestovat konkrétní členskou metodu třídy, která se o ukládání dat do cache[40] starala. Unit testy jsou tedy velice užitečným nástrojem při vývoji softwaru. Osobně bych se bez nich při vývoji rozsáhlých aplikací již neobešel.

[37] Zlepšení existujícího kódu
[38] Integrovaný dotazovací jazyk
[39] Kolekce obrázků zahrnuta v jediném obrázku
[40] Vyrovnávací paměť

7 Závěr

7.1 Uplatněné znalosti získané studiem

Dle mého názoru jsem praktické znalosti a dovednosti získané v průběhu studia uplatnil v rámci odborné praxe pouze částečně. Většinu zkušeností s tvorbou webových aplikací jsem získal až v průběhu praxe a to vesměs ve formě samostudia. Samotné studium mi poskytlo spíše určité teoretické základy, díky kterým jsem nad řešenými úkoly dokázal přemýšlet více v souvislostech. Což mi následně pomohlo ke snadnější orientaci při práci a také k dosažení lepší efektivity při vývoji. Za užitečné považuji především předměty *Databázové a informační systémy*, *Vývoj internetových aplikací*, *Informační systémy pro elektronické podnikání* a *Architektura technologie .NET*. Naopak za méně přínosné považuji předměty z oblasti elektroniky a telekomunikací, které jsem využil absolutně minimálně.

Nicméně velice důležitá byla i znalost anglického jazyka, jelikož jsem se značnou část praxe věnoval právě překladům do angličtiny (v rámci vývoje webové aplikace nazývané CookBook). Osobně beru jako přínos i zkušenosti z volitelného předmětu *Uživatelská rozhraní*, kde jsem nabyl povědomí o tvorbě designu těchto rozhraní.

7.2 Scházející znalosti

Jelikož jsem část praxe absolvoval ještě před zahájením třetího ročníku bakalářského studia, chyběla mi celá řada znalostí a dovedností. Zejména znalost návrhových vzorů a technologie .NET, se kterou jsem se v rámci studia seznámil až v šestém semestru. Dále mi chyběla praktická zkušenost s vývojem reálných aplikací, kterou šlo získat pouze při vlastní tvorbě semestrálních projektů. Kromě toho jsem postrádal i jakoukoliv zkušenost se sdíleným zdrojovým kódem (narážím zde na verzovací systém TFS), psaním unit testů a XSL transformacemi.

7.3 Zhodnocení odborné praxe

Z mého pohledu byla odborná praxe ve společnosti NetDirect velmi přínosnou zkušeností. Díky práci na vývojovém oddělení jsem měl možnost aplikovat nabyté vědomosti na reálných projektech, čímž jsem přišel do styku s celou řadou technologií a vývojových metod. Řešení zadaných úkolů mě postupně donutilo naučit se celou řadu nových poznatků, jež obohatily můj budoucí profesní život. Jako jeden z největších přínosů vnímám zkušenost s prací v týmu, kdy jsem si mohl zcela poprvé osobně vyzkoušet agilní metodiku Scrum a také práci s verzovacím systémem TFS. Rovněž jsem si mohl vyzkoušet i práci na rozsáhlém a komplexním řešení, čímž jsem získal ucelenější pohled, jak daný vývoj funguje a co je při něm vše zapotřebí. Kromě toho jsem se naučil i správně využívat unit testování, které mi do té doby přišlo zcela zbytečné.

Dosažené výsledky za dobu mé odborné praxe tedy hodnotím velice kladně. A osobně jsem velmi rád, že jsem se rozhodl absolvovat odbornou praxi právě ve společnosti NetDirect.

8 Reference

[1] ERL, Thomas. *SOA design pattern*. 1st ed. Indianapolis: Prentice Hall, c2009. ISBN 978-0-13-613516-6.

[2] SHARP, John. *Microsoft Windows Communication Foundation step by step*. Redmond, Wash.: Microsoft Press, c2007. ISBN 07-356-2336-8.

[3] BATTAT, Suliman. *Agile Project Management using TFS 2012*. [online]. [cit. 2013-05-17]. Dostupné z: `http://msdn.microsoft.com/en-us/magazine/dn189203.aspx`

[4] CHAPPELL, David. *Understanding .NET*. 2nd ed. Upper Saddle River, NJ: Addison-Wesley, xv, 317 p. ISBN 978-032-1194-046.

[5] NAGEL, Christian. *Professional C# 4 and .Net 4*. Indianapolis, IN: Wiley Pub., c2010. ISBN 04-705-0225-8.

[6] PALERMO, Jeffrey. *ASP.NET MVC 4 in action*. Revised edition of ASP.NET MVC 2 in action. Manning Publications, 2012, 406 pages. Third Edition. ISBN 16-172-9041-6.

[7] NOVATCHEV, Dimitre. *XSLT 2.0 and 1.0 Foundations*. [online]. [cit. 2014-02-19]. Dostupné z: `http://www.pluralsight.com/training/Courses/TableOfContents/xslt-foundations-part1`